Internet Engineering Task Force
INTERNET-DRAFT
draft-ietf-dccp-ccid3-03.ps

Sally Floyd
Eddie Kohler
ICIR
Jitendra Padhye
Microsoft Research

30 June 2003
Expires: December 2003

Profile for DCCP Congestion Control ID 3: TFRC Congestion Control

Status of this Document

Abstract

This document contains the profile for Congestion Control Identifier 3, TCP-friendly rate control (TFRC), in the Datagram Congestion Control Protocol (DCCP). DCCP implements a congestion-controlled unreliable datagram flow suitable for use by applications such as streaming media. The TFRC CCID is used by applications that want a TCP-friendly send rate, possibly with Explicit Congestion Notification (ECN), while minimizing abrupt rate changes.

TO BE DELETED BY THE RFC EDITOR UPON PUBLICATION:

Changes from draft-ietf-dccp-ccid3-02.txt:

* Added to the section on Application Requirements.

* Added a section on Packet Sizes.

Changes from draft-ietf-dccp-ccid3-01.txt:

* Added "Security Considerations" and "IANA Considerations" sections.

* Store Window Counter in the DCCP header's CCval field, not a separate option.

* Add to the description of a loss interval in the Loss Intervals option: a loss interval includes at most one round-trip time's worth of possibly-marked packets, and at least one round-trip time's worth of packets in all.

* Added a description of when the loss event rate calculated by the sender could differ from that calculated by the receiver.

* Window counter fixups.

* Add Use Loss Intervals and Use Loss Event Rate features, and explain their interaction.

* Move Elapsed Time option to DCCP's main specification (and simultaneously change its units to tenths of milliseconds). Allow the use of either Elapsed Time or Timestamp Echo.

* Clarify the definition of quiescence.

* Change calculations for determining loss events to take window counter wrapping into account.

Changes from draft-ietf-dccp-ccid3-00.txt:

* Changed the guidelines to say that required acknowledgement packets should include one or more of the following: The Loss Event Rate, Loss Intervals, or the Ack Vector.

* Added a separate section on "The Use of Ack Vectors". This section says that Ack-of-acks must be used when the Ack Vector is used.

* Renamed the "ECN Nonce Option" to the "Loss Intervals" option, and extended this option to include up to eight loss intervals. This is to enable more precise verification by the sender of the receiver's feedback.

* Added a section about "When should Ack Vector or Loss Intervals be used?" In progress.

* Added a section about using the ECN Nonce to verify the receiver's feedback.

* Said that the ECN-Nonce feedback must be returned in every required acknowledgement.

* Added a sentence saying that the TFRC spec "separately specifies the minimum sending rate from rate reductions during an idle period."

Table of Contents

1. Introduction

This document contains the profile for Congestion Control Identifier 3, TCP-friendly rate control (TFRC), in the Datagram Congestion Control Protocol (DCCP). DCCP uses Congestion Control Identifiers, or CCIDs, to specify the congestion control mechanism in use on a half-connection. (A half-connection might consist of data packets sent from DCCP A to DCCP B, plus acknowledgements sent from DCCP B to DCCP A. DCCP A is the HC-Sender, and DCCP B the HC-Receiver, for this half-connection. In this document, we abbreviate HC-Sender and HC-Receiver as "sender" and "receiver", respectively. These terms are defined more fully in [DCCP].)

TFRC is a receiver-based congestion control mechanism that provides a TCP-friendly send rate, while minimizing abrupt rate changes [RFC 3448].

The basic TFRC protocol is as follows. The sender sends a stream of data packets to the receiver at some rate. The receiver sends a feedback packet to the sender roughly once every round-trip time. Based on the information contained in the feedback packets, the sender adjusts its sending rate in accordance with the TCP throughput equation [PFTK98], to maintain TCP-friendliness. If no feedback is received from the receiver in several round-trip times (four, in the current TFRC specification), the sender halves its sending rate.

The values of the round-trip time *RTT*, the loss event rate *p* and the base timeout value *TO* are needed by the sender to calculate the send rate using the TCP throughput equation. The sender calculates the values of *RTT* and *TO*, and the receiver calculates the value of *p*. (If it prefers, the sender can also calculate *p* based on loss intervals provided by the receiver.)

The congestion control mechanisms described here follow the TFRC mechanism standardized by the IETF. Conformant CCID 3 implementations MAY track TFRC's evolution directly, as updates are standardized in the IETF, rather than waiting for revisions of this document.

For simplicity, we occasionally refer to DCCP-Data packets sent by the sender and DCCP-Ack packets sent by the receiver. Both of these categories are meant to include DCCP-DataAck packets.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119].

1.1. Usage Scenario

DCCP with TFRC congestion control is intended to provide congestion control for the flow of data packets from the server to the client for applications that do not require fully reliable data transmission, or that desire to implement reliability on top of DCCP. TFRC congestion control is appropriate for flows that would prefer to minimize abrupt changes in the sending rate.

1.2. Example Half-Connection

This example shows the typical progress of a half-connection using TFRC Congestion Control specified by CCID 3, not including connection initiation and termination. Again, the "sender" is the HC-Sender, and the "receiver" is the HC-Receiver. (The example is informative, not normative.)

(1) The sender sends DCCP-Data packets, where the number of packets sent is governed by an allowed transmit rate, as specified in [RFC 3448]. Each DCCP-Data packet has a sequence number, and the DCCP header's CCval field contains the window counter value.

One or more of these data packets are DCCP-DataAck packets acknowledging the data packet from the receiver, but for simplicity we will not discuss the half-connection of data from the receiver to the sender in this example.

If the use of ECN has been negotiated, each DCCP-Data and DCCP-DataAck packet is sent as ECN-Capable, with either the ECT(0) or the ECT(1) codepoint set. The use of the ECN Nonce with TFRC is described below.

(2) The receiver sends DCCP-Ack packets at least once per round-trip time acknowledging the data packets, unless the sender is sending at a rate of less than one packet per RTT, as indicated by the TFRC specification [RFC 3448]. Each DCCP-Ack packet uses a sequence number and identifies the most recent packet received from the sender. Each DCCP-Ack packet includes feedback about the loss event rate calculated by the receiver, as specified below.

(3) The sender continues sending DCCP-Data packets as controlled by the allowed transmit rate. Upon receiving DCCP-Ack packets, the sender updates its allowed transmit rate as specified in [RFC 3448].

(4) The sender estimates round-trip times and calculates a TimeOut value *TO* as specified in [RFC 3448].

2. Connection Establishment

The connection is initiated by the client using mechanisms described in the DCCP specification [DCCP]. During or after CCID 3 negotiation, the client and/or server MAY want to negotiate the values of the Use Ack Vector, Use Loss Intervals, and Use Loss Event Rate features.

3. Congestion Control on Data Packets

The sender sends DCCP-Data packets to the receiver at the rate specified by the TCP throughput equation [PFTK98].

Each DCCP-Data packet has a sequence number and, in the DCCP header's CCval field, a window counter value. The window counter is described below.

After each feedback packet is received from the receiver, the sender updates values of *RTT*, *TO* and the sending rate using procedures specified in [RFC 3448].

If no feedback packet is received from the receiver after an interval specified in [RFC 3448], the sending rate is halved. However, the sending rate is never reduced below one packet per 64 seconds. See [RFC 3448] for more details. [RFC 3448] separately specifies the minimum sending rate from rate reductions during an idle period.

4. Acknowledgements

The receiver sends an acknowledgement packet to the sender roughly once per round-trip time, if the sender is sending packets that frequently. This rate is determined by details of the TFRC protocol, as specified in [RFC 3448].

As specified in [DCCP], the acknowledgement number acknowledges the greatest valid sequence number received so far on this connection. ("Greatest" is, of course, measured in circular sequence space.) Each acknowledgement required by TFRC also includes at least the following options:

(1) An Elapsed Time and/or Timestamp Echo option specifying the amount of time elapsed since the receiver received the packet whose sequence number appears in the Acknowledgement Number field.

(2) A Receive Rate option specifying the rate at which the receiver received data since the last DCCP-Ack was sent.

(3) One or more options concerning the loss event rate *p* experienced by the receiver, as described in [RFC 3448]. Relevant options include Loss Event Rate, which simply gives the loss event rate calculated by the receiver; Loss Intervals, which specifies the beginning and end of each loss interval, from which the sender can easily calculate and/or verify the loss event rate; and Ack Vector, which says exactly which packets were lost or marked, again allowing the sender to calculate and/or verify the loss event rate.

The format of these options is described below (except Ack Vector, Timestamp Echo, and Elapsed Time, which are described in [DCCP]).

If the HC-Receiver is also sending data packets to the HC-Sender, then it MAY piggyback acknowledgement information on those data packets more frequently than TFRC's specified acknowledgement rate allows.

4.1. Congestion Control on Acknowledgements

The rate and timing for generating acknowledgements is determined by the TFRC algorithm [RFC 3448]. The sending rate for acknowledgements is relatively low, and there is no explicit congestion control on the acknowledgements.

4.2. Quiescence

This section refers to quiescence in the DCCP sense (see section 8.1 of [DCCP]): How does a CCID 3 receiver determine that the corresponding sender is not sending any data?

Let T equal the greater of 0.2 seconds and two round-trip times. The receiver detects that the sender has gone quiescent after T seconds have passed without receiving any additional data from the sender.

4.3. Acknowledgements of Acknowledgements

TFRC acknowledgements are not generally required to be reliable, so the sender generally need not acknowledge the receiver's acknowledgements. When Ack Vector is used, however, the sender, DCCP A, MUST occasionally acknowledge the receiver's acknowledgements so that the receiver can free up Ack Vector state. When both half-connections are active, the necessary acknowledgements will be contained in A's acknowledgements to B's data. If the B-to-A half-connection goes quiescent, however, DCCP A must do it proactively.

When Ack Vector is used, therefore, an active sender MUST occasionally acknowledge the receiver's acknowledgements, probably by encapsulating a datagram in a DCCP-DataAck packet. No acknowledgement options are necessary, just the relevant Acknowledgement Number in the DCCP-DataAck header. Such acknowledgements should be sent approximately once per round-trip time, within a factor of two or three.

The sender MAY choose to acknowledge the receiver's acknowledgements even if they do not contain Ack Vectors. For instance, regular acknowledgements can shrink the size of the

Loss Intervals option. Unlike the Ack Vector, however, the Loss Intervals option is bounded in size (and receiver state), so acks-of-acks are not required.

5. Explicit Congestion Notification

ECN [RFC 3168] MAY be used with CCID 3. If ECN is enabled, then the ECN Nonce will automatically be used following the specification for the ECN Nonce for TCP [ECN NONCE]. For the data sub-flow, the sender sets either the ECT[0] or ECT[1] codepoint on DCCP-Data packets.

If ECN is used, then the receiver MUST use at least one of Ack Vector and Loss Intervals to return ECN Nonce information to the sender.

If the Ack Vector option is being used, the ECN nonce sum is returned in DCCP-Ack packets, as described in [CCID 2 PROFILE]. The sender can maintain a table with the ECN nonce sum for each packet, and use this information to probabilistically verify the ECN nonce sum returned in each DCCP-Ack packet.

If the Ack Vector option is not being used, the information about the ECN Nonce is returned by the receiver using the Loss Intervals option described below. The receiver MUST include this option on every required acknowledgement.

6. Relevant Options and Features

CCID 3 can make use of DCCP's Ack Vector, Timestamp, Timestamp Echo, and Elapsed Time options and its Use Ack Vector and ECN Capable features. In addition, the following CCID-specific values, options, and features are defined for use with CCID 3.

The use of Ack Vector, Loss Intervals, and Loss Event Rate are controlled by separate features, but only some combinations of these features make sense. In particular, if ECN Capable is true, then every required acknowledgement MUST include at least one of Ack Vector and Loss Intervals; otherwise, every required acknowledgement MUST include at least one of Ack Vector, Loss Intervals, and Loss Event Rate. This may impel the receiver to send certain options even when their corresponding Use features are false. A sender that receives several invalid acknowledgements---that include only Loss Event Rate on an ECN-capable connection, for example---MAY respond by resetting the connection with Reason set to "Option Error".

6.1. Window Counter Value

The data sender stores a 4-bit window counter value in the DCCP generic header's CCval field on every data packet it sends. This value is set to 0 at the beginning of the transmission, and generally increased by 1 every quarter of a round-trip time, as described in [RFC 3448]. For reference, the DCCP generic header is as follows (diagram repeated from [DCCP]):

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |           Dest Port           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Type  | CCval |              Sequence Number                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data Offset  | # NDP | Cslen |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The CCval field has enough space to express 4 round-trip times at quarter-RTT granularity. The sender SHOULD try to avoid wrapping CCval on adjacent packets, as might happen, for example, if two data-carrying packets were sent 4 round-trip times apart with no packets intervening. For example, the sender MAY use the following algorithm for setting CCval. The algorithm uses three variables: "last_WC" holds the last window counter value sent, "last_WC_time" is the time at which the first packet with window counter value "last_WC" was sent, and "RTT" is the current round-trip time estimate. last_WC is initialized to zero, and last_WC_time to the time of the first packet sent. Then, before sending a new packet, proceed like this:

```
Let quarter_RTTs = floor( (current_time - last_WC_time) / (RTT/4) ).
If quarter_RTTs > 0, then:
    Set last_WC := (last_WC + min(quarter_RTTs, 5)) mod 16, and
    Set last_WC_time := current_time.
Set the packet header's CCval field to last_WC.
```

The window counter value may also change as feedback packets arrive. In particular, after receiving an acknowledgement for a packet sent with window counter WC, the sender SHOULD increase its window counter, if necessary, so that subsequent packets have window counter value at least (WC + 4) mod 16.

6.2. Elapsed Time Options

The data receiver MUST include an elapsed time value on every required acknowledgement. This helps the sender distinguish between network round-trip time, which it must include in its rate equations, and delay at the receiver due to TFRC's infrequent acknowledgement rate. The elapsed time value MUST be included in one of two ways:

(1) If at least one recent data packet (i.e., a packet received after the previous DCCP-Ack was sent) included a Timestamp option, then the receiver SHOULD include the corresponding Timestamp Echo option, with Elapsed Time value.

(2) Otherwise, the receiver MUST include an Elapsed Time option.

All these option types are defined in the main DCCP specification [DCCP].

6.3. Receive Rate Option

```
+--------+--------+--------+--------+--------+--------+
|11000010|00000110|            Receive Rate           |
+--------+--------+--------+--------+--------+--------+
 Type=194  Len=6
```

This option MUST be sent by the data receiver on all required acknowledgements. The first byte gives the option type and the second gives the option length. The last four bytes indicate the rate at which the receiver has received data since it last sent an acknowledgement, in bits per second.

6.4. Use Loss Event Rate Feature

The Use Loss Event Rate feature lets CCID 3 endpoints negotiate whether the receiver MUST provide Loss Event Rate options on its acknowledgements.

Use Loss Event Rate has feature number 192. The Use Loss Event Rate feature located at DCCP B specifies whether DCCP B MUST send Loss Event Rate options on its acknowledgements, although DCCP B MAY send Loss Event Rate options even if Use Loss Event Rate is false. DCCP A sends a "Change(Use Loss Event Rate, 1)" option to ask DCCP B to send Loss Event Rate options as part of its acknowledgement traffic.

Use Loss Event Rate feature values are a single byte long. The receiver MUST send Loss Event Rate options if this byte is nonzero. A CCID 3 half-connection starts with Use Loss Event Rate unknown.

6.5. Loss Event Rate Option

```
+--------+--------+--------+--------+--------+--------+
|11000000|00000110|          Loss Event Rate          |
+--------+--------+--------+--------+--------+--------+
 Type=192   Len=6
```

The option value indicates the inverse of the loss event rate, rounded UP, as calculated by the receiver. Its units are packets per loss interval.

6.6. Use Loss Intervals Feature

The Use Loss Intervals feature lets CCID 3 endpoints negotiate whether the receiver MUST provide Loss Intervals options on its acknowledgements.

Use Loss Intervals has feature number 195. The Use Loss Intervals feature located at DCCP B specifies whether DCCP B MUST send Loss Intervals options on its acknowledgements, although DCCP B MAY send Loss Intervals options even if Use Loss Intervals is false. DCCP A sends a "Change(Use Loss Intervals, 1)" option to ask DCCP B to send Loss Intervals options as part of its acknowledgement traffic.

Use Loss Intervals feature values are a single byte long. The receiver MUST send Loss Intervals options if this byte is nonzero. A CCID 3 half-connection starts with Use Loss Intervals unknown.

6.7. Loss Intervals Option

```
                    ___ Loss Interval ___
                   /                     \
+--------+--------+----...----+----...----+--------+--------+--------
|11000011| Length | Left Edge |E|  Offset | Up to 7 Loss Intervals ...
+--------+--------+----...----+----...----+--------+--------+--------
 Type=195            3 bytes     3 bytes
```

This option MAY be set by the data receiver on acknowledgements. (If ECN is enabled and Ack Vector is off, or if the Use Loss Intervals feature is true, it MUST be sent with every required acknowledgement.) The option reports up to 8 loss intervals seen by the receiver.

As described in [RFC 3448], a loss interval begins with a lost or ECN-marked packet; continues with at most one round trip time's worth of packets that may or may not be lost or marked; and completes with an arbitrarily-long series of non-dropped, non-marked packets. In addition, as specified in [RFC 3448], a loss interval continues for at least one round trip time; a lost or marked packet starts a new loss interval only if it was sent at least one round trip time after the start of the previous loss interval. The Loss Event Rate, reported by option 192, is the weighted average of the last 8 loss interval lengths, inverted.

The Loss Intervals option contains information about between one and eight consecutive loss intervals, always including the most recent loss interval. Intervals are listed in reverse chronological order. The option MUST contain information about the most recent 8 loss intervals unless (1) there have not yet been 8 loss intervals, in which case the receiver SHOULD send information about all the loss intervals it has experienced; or (2) the receiver knows, because of acknowledgements from the sender, that information about older loss intervals has been received by the sender, in which case the receiver MUST send at least information about the loss intervals the sender has not acknowledged. In any case, the Loss Intervals option MUST contain the most recent loss interval.

Each Loss Interval structure consists of a Left Edge, an Offset, and an ECN Nonce Echo (E). Left Edge, a 24-bit DCCP sequence number, specifies the first sequence number in the interval's loss- and mark-free tail. Offset, a 23-bit number, specifies the number of packets in that loss- and mark-free tail. The ECN Nonce Echo, stored in the high-order bit of the 3-byte field containing Offset, equals the one-bit sum (exclusive-or, or parity) of nonces received over the range of packets [Left Edge, Left Edge + Offset). If Offset is 0, or if the receiver is ECN-incapable, the ECN Nonce Echo SHOULD be reported as 0.

Note that each Loss Interval structure explicitly specifies when the loss interval in question *ends* (that is, at Left Edge + Offset), but not when it *began*. That quantity equals the Left Edge + Offset of the chronologically preceding loss interval. Furthermore, the most recent Loss Interval's Left Edge + Offset need not equal the Acknowledgement Number. As Section 5.1 of [RFC 3448] says, a lost packet doesn't begin a new loss interval until 3 packets have been seen after the "hole". Acknowledgements sent in the meantime will acknowledge some sequence number larger than the "hole", but the most recent Loss Interval's Left Edge + Offset will equal the sequence number of the "hole".

The Loss Intervals option serves several purposes.

- The sender can use the Loss Intervals to easily calculate the Loss Event Rate, perhaps using a later version of the TFRC algorithm than that deployed at the receiver.
- Loss Intervals information is easily checked for consistency against previous Loss Intervals options, and against any Loss Event Rate calculated by the receiver.
- The sender can probabilistically verify the ECN Nonce Echo for each Loss Interval, reducing the likelihood of misbehavior.

7. Verifying Congestion Control Compliance With ECN

If ECN is used, the sender can use Ack Vector or the Loss Intervals option to probabilistically verify that the receiver is not lying in reporting packets received undropped and unmarked. The sender could then use the information in acknowledgement packets to roughly verify the Loss Event Rate reported by the receiver, if it so desired.

We note that if ECN is not used, the sender could still check on the receiver by occasionally not sending a packet, or sending a packet out-of-order, to catch the receiver in an error in Ack Vector or Loss Intervals information. Similarly, the sender would still use the Ack Vector or Loss Intervals information to verify the loss event rate reported by the receiver. However, this is not as robust or as non-intrusive as the verification provided by the ECN Nonce.

7.1. Verifying the ECN Nonce Echo

To verify the ECN Nonce Echo included with an Ack Vector option, the sender maintains a table with the ECN nonce value sent for each packet. The Ack Vector option explicitly says which packets were received non-marked; the sender just adds up the nonces for those packets using a one-bit sum (exclusive-or, or parity), and compares the result to the Nonce Echo encoded in the Ack Vector's option type.

To verify the ECN Nonce Echo included with a Loss Intervals option, the sender maintains a table with the ECN nonce *sum* for each packet. As defined in [ECN NONCE], the nonce sum for sequence number S is the one-bit sum of nonces over the sequence number range [I,S] (where I is the initial sequence number). Let NonceSum(S) represent this nonce sum for sequence number S, and let NonceSum(I - 1) equal 0. Then the Nonce Echo for a loss interval [Left Edge, Left Edge + Offset) should equal the following one-bit sum:

NonceSum(Left Edge - 1) + NonceSum(Left Edge + Offset - 1).

An Ack Vector's ECN Nonce Echo may also be calculated from a table of ECN nonce sums, rather than ECN nonces. If the Ack Vector contains many long runs of non-marked, non-dropped packets, the nonce sum-based calculation will probably be faster than a straightforward nonce-based calculation.

In either of these cases, a misbehaving receiver---meaning a receiver that reports a lost or marked packet as "received non-marked", to avoid rate reductions---has only a 50% chance of guessing the correct Nonce Echo.

7.2. Verifying the Reported Loss Event Rate

Once the sender has probabilistically verified the ECN Nonce Echoes reported by the receiver, the sender can calculate for itself the number of packets in each loss interval, to roughly verify the loss event rate reported by the receiver, if it so desires. We note that DCCP's Loss Event Rate Option reports the average loss interval size, which is the inverse of the loss event rate.

If the Ack Vector is used, the sender can identify the packet that begins each new loss interval from the Ack Vector in each DCCP-Ack packet. If the sender saves information about the window counter for each data packet, then the sender also can tell when two lost or marked packets would have been interpreted by the receiver as separate loss events.

The Loss Intervals option explicitly reports the size of each loss interval, as seen by the receiver. The sender can, using saved information about window counters, verify that the receiver is not falsely combining two loss events into one reported loss interval.

Once the sender has reconstructed or verified Loss Intervals, it can easily calculate the expected loss event rate, and compare against the receiver's reported loss event rate.

We note that in some cases the loss event rate calculated by the sender could differ from that calculated by the receiver. In particular, when a number of successive packets are dropped, the receiver does not know the sending times for these packets, and interprets these losses as a single loss event. In contrast, if the sender has saved the sending times or the window counter information for these packets, then the sender can determine if these losses constitute a single loss event, or several successive loss events. Thus, with its knowledge of the sending times of dropped packets, the sender is able to make a more accurate calculation of the loss event rate.

8. Application Requirements

CCID 3 is appropriate for flows that would prefer to minimize abrupt changes in the sending rate. Applications that prefer a relatively smooth sending rate include some streaming media applications with small or moderate buffering at the receive application before the playback time. TCP-like congestion control, which halves the sending rate in response to a congestion event, cannot satisfy this preference for a relatively smooth sending rate.

As explained in [RFC 3448], the penalty of having smoother throughput than TCP while competing fairly for bandwidth is that the TFRC mechanism in CCID 3 responds slower than TCP or TCP-like mechanisms to changes in available bandwidth. Thus CCID 3 should only be used when the application has a requirement for smooth throughput, in particular, avoiding TCP's halving of the sending rate in response to a single packet drop. For applications that simply need to transfer as much data as possible in as short a time as possible we recommend using TCP-like congestion control.

As described in the TFRC specifications [RFC 3448], this CCID should also not be used by applications that change their sending rate by varying the packet size, rather than varying the rate at which packets are sent. A new CCID will be required for these applications.

9. Design Considerations

CCID 3 data packets need not carry Timestamp options. The sender can store the times at which recent packets were sent. Then the Acknowledgement Number and Elapsed Time option contained on each required acknowledgement provide sufficient information to compute the round trip time. Alternatively, the sender MAY include Timestamp options on a limited subset of its data packets; the receiver will respond with Timestamp Echo options including Elapsed Times, allowing the sender to calculate round-trip times without storing timestamps at all.

9.1. Determining Loss Events at the Receiver

The window counter is used by the receiver to determine if multiple lost packets belong to the same loss event. The sender increases the window counter by 1 every quarter round trip time. To determine whether two lost packets, with sequence numbers X and Y (Y > X in circular sequence space), belong to different loss events, the receiver proceeds as follows:

- Let X_prev be the greatest sequence number which was received with X_prev < X.
- Let Y_prev be the greatest sequence number which was received with Y_prev < Y.
- Given a sequence number N, let C(N) be the window counter value associated with that packet.
- Packets X and Y belong to different loss events if there exists a packet with sequence number S so that X_prev < S <= Y_prev, and the distance from C(X_prev) to C(S) is greater than 4. (The distance is the number D so that C(X_prev) + D = C(S) (mod WCTRMAX), where WCTRMAX is the maximum value for the window counter---in our case, 16.)

  This complex calculation is necessary to handle the case where window counter space wrapped completely between X and Y. Generally, the receiver can simply check whether the distance from C(X_prev) to C(Y_prev) is greater than 4.

Window counters can help the receiver to disambiguate multiple losses after a sudden decrease in the actual round-trip time. When the sender receives an acknowledgement acknowledging a data packet with window counter *i*, the sender increases its window counter, if necessary, so that subsequent data packets are sent with window counter values of at least *i*+4. This can help minimize errors on the part of the receiver of incorrectly interpreting multiple loss events as a single loss event.

We note that if all of the packets between X and Y are lost in the network, then X_prev and Y_prev are both set to X-1, and the series of consecutive losses is treated by the receiver as a single loss event. However, the sender will receive no DCCP-Ack packets during a period of consecutive losses, and the sender will reduce its sending rate accordingly.

As an alternative to the window counter, the sender could have sent its estimate of the round-trip time to the receiver directly in a round-trip time option, and the receiver should use the sender's round-trip time estimate to infer when multiple lost or marked packets belong in the same loss event. In some respects, a round-trip time option gives a more precise encoding of the sender's round-trip time estimate than does the window counter. However, the window counter conveys information about the relative *sending* times for packets, while the receiver could only use the round-trip time option to distinguish between the relative *receive* times (in the absence of timestamps). That is, the window counter will give more robust performance in some cases when there is a large variation in delay for packets sent within a window of data. As a slightly more speculative consideration, the round-trip time option could possibly be used more easily by middleboxes attempting to verify that a flow was using conformant end-to-end congestion control.

9.2. Sending Feedback Packets

The window counter is also used by the receiver to decide when to send feedback packets. Feedback packets should normally be sent at least once per round-trip time, if the sender is sending at least one data packet per round-trip time. Whenever the receiver sends a feedback message, the receiver sets a local variable last_counter to the greatest received value of the window counter since the last feedback message was sent, if any data packets have been received since the last feedback message was sent. If the receiver receives a data packet with a window counter value greater than or equal to last_counter + 4, then the receiver sends a new feedback packet. ("Greater" and "greatest" are measured in circular window counter space.)

The TFRC protocol [RFC 3448] specifies that the receiver uses a feedback timer to decide when to send feedback packets. In the TFRC protocol, when the feedback timer expires, the receiver resets the timer to expire after R_m seconds, where R_m is the most recent estimate of the round-trip time received by the receiver from the sender. However, when the window counter is used, the receiver can use its information in deciding when to send feedback packets.

When the sender is sending less than one packet per round-trip time, then the receiver sends a feedback packet after each data packet, and the feedback timer is not required. Similarly, when the sender is sending several packets per round-trip time, then the receiver will send a feedback packet each time that a data packet arrives with a window counter more than four greater than the window counter when the last feedback packet was sent, and again the feedback counter is not required. Similarly, the receiver always sends a feedback packet after the detection of a loss event. Thus, the feedback timer is not absolutely necessary when the window counter is used.

However, the feedback timer still could be useful in some rare cases to prevent the sender from unnecessarily halving its sending rate. Consider the case when the receiver receives data soon after the most recent feedback packet has been sent, but has received no data packets with a window counter sufficiently large to trigger sending a new feedback packet. The TFRC protocol specifies that after a feedback packet is received, the sender sets a nofeedback timer to at least four times the round-trip time estimate. If the sender doesn't receive any feedback packets before the nofeedback timer expires, then the sender halves its sending rate. One could construct scenarios where the use of a feedback timer at the receiver would prevent the unnecessary expiration of the nofeedback timer at the sender.

For implementors who wish to implement a feedback timer for the data receiver, we suggest estimating the round-trip time from the most recent data packet as follows: Let K be the window counter from the most recent data packet, and let T_k be the time that that packet was received, as in the table below. Let J be the highest window counter received that was less than K-4, and let T_j be the most recent time that such a packet was received. Then the round-trip time can be very roughly estimated as 4*(T_k-T_j)/(K-J).

Time	Event	Window Counter
T_j	packet received with WC < K-4	J (J<K-4)
T_k	most recent packet received	K

9.3. When Should Ack Vector And Loss Intervals Be Used?

If the use of ECN has not been negotiated, then the receiver is not required to use either Ack Vector or Loss Intervals. Essentially, in this case the sender is completely relying on the Loss Event Rate reported by the receiver. If the Ack Vector or Loss Intervals is used, however, then the sender could test that the receiver is correctly reporting dropped and marked packets by conducting a test and skipping a packet in its transmissions.

In the common case, it is assumed that the use of ECN will be negotiated with CCID 3. However, it is possible that either the sender or the receiver will want to negotiate the use of CCID 3 without ECN, e.g., if there happens to be a known broken middlebox along the path that blocks the use of ECN in the IP packet header.

If ECN is used, then the receiver is required to use at least one of Ack Vector and Loss Intervals to return ECN Nonce information to the sender. The Ack Vector returns more information about which packets were lost or marked during a loss event. The sender uses more computation and state for verifying receiver feedback with the Ack Vector than with Loss Intervals, because then it must reconstruct loss intervals from the Ack Vector. The Ack Vector also requires that the sender occasionally acknowledge the receiver's acknowledgements; this is optional with Loss Intervals.

9.4. Packet Sizes

CCID 3 is intended for applications that use a fixed packet size, and that vary their sending rate in packets per second in response to congestion. CCID 3 is not appropriate for applications that require a fixed interval of time between packets, and vary their packet size instead of their packet rate in response to congestion. However, some attention might be required for applications using CCID 3 that vary their packet size not in response to congestion, but in response to other application-level requirements.

10. Thanks

We thank Mark Handley for his help in defining CCID 3. We thank Sara Karlberg, Arun Venkataramani, and Yufei Wang for feedback on earlier versions of this document.

11. Normative References

[CCID 2 PROFILE] S. Floyd and E. Kohler. Profile for DCCP Congestion Control ID 2: TCP-like Congestion Control, draft-ietf-dccp-ccid2-01.txt, work in progress, March 2003.

[DCCP] E. Kohler, M. Handley, S. Floyd, and J. Padhye. Datagram Congestion Control Protocol, draft-ietf-dccp-spec-01.txt, work in progress, March 2003.

[ECN NONCE] Neil Spring, David Wetherall, and David Ely. Robust ECN Signaling with Nonces, draft-ietf-tsvwg-tcp-nonce-04.txt, work in progress, October 2002.

[RFC 2119] S. Bradner. Key Words For Use in RFCs to Indicate Requirement Levels. RFC 2119.

[RFC 3168] K.K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168. September 2001.

[RFC 3448] M. Handley, S. Floyd, J. Padhye, and J. Widmer, TCP Friendly Rate Control (TFRC): Protocol Specification, RFC 3448, Proposed Standard, January 2003.

12. Informative References

[PFTK98] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. Proc ACM SIGCOMM 1998.

13. Security Considerations

Security considerations for DCCP have been discussed in [DCCP], and security considerations for TFRC have been discussed in [RFC 3448]. The security considerations for TFRC include the need to protect against spoofed feedback, and the need for protection mechanisms to protect the congestion control mechanisms against incorrect information from the receiver.

In this document we have extensively discussed the mechanisms the sender can use to verify the information sent by the receiver.

14. IANA Considerations

This section will contain the namespaces that have been created in this specification, and the values assigned in existing namespaces managed by IANA.

This will include the following: The Receive Rate, Loss Event Rate, and Loss Intervals Options; the Use Loss Event Rate and Use Loss Intervals features.

15. Authors' Addresses

Sally Floyd <floyd@icir.org>
Eddie Kohler <kohler@icir.org>

ICSI Center for Internet Research
1947 Center Street, Suite 600
Berkeley, CA 94704 USA

Jitendra Padhye <padhye@microsoft.com>

Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA